Journal of information and communication convergence engineering
http://jicce.org

J. Inf. Commun. Converg. Eng. 11(2): 118-123, Jun. 2013 **Regular paper**

An Efficient Interpolation Hardware Architecture for HEVC Inter-Prediction Decoding

Xianzhe Jin and Kwangki Ryoo*, *Member, KIICE*

Graduate School of Information and Communication, Hanbat National University, Daejeon 305-719, Korea

Abstract

This paper proposes an efficient hardware architecture for high efficiency video coding (HEVC), which is the next generation video compression standard. It adopts several new coding techniques to reduce the bit rate by about 50% compared with the previous one. Unlike the previous H.264/AVC 6-tap interpolation filter, in HEVC, a one-dimensional seven-tap and eight-tap filter is adopted for luma interpolation, but it also increases the complexity and gate area in hardware implementation. In this paper, we propose a parallel architecture to boost the interpolation performance, achieving a luma 4×4 block interpolation in 2–4 cycles. The proposed architecture contains shared operations reducing the gate count increased due to the parallel architecture. This makes the area efficiency better than the previous design, in the best case, with the performance improved by about 75.15%. It is synthesized with the MagnaChip 0.18 μm library and can reach the maximum frequency of 200 MHz.

Index Terms: HEVC decoding, Inter-predictor, Interpolation, Parallel architecture

I. INTRODUCTION

High efficiency video coding (HEVC) [1], which will be finalized in 2013, is the most recent joint video project of the Joint Collaborative Team on Video Coding (JCT-VC). The purpose of HEVC is to save the bit-rate by about 50% compared with the previous H.264/AVC standard [2], and it is targeted for up to ultra high-definition (UHD, 7680×4320) resolution. For the previous H.264/AVC standard, a number of studies have been performed, such as [3-5]. However, the research for HEVC remains insufficient, especially regarding hardware design.

HEVC allows for various new coding techniques, such as 35 luma intra-prediction modes, spatial and temporal MV prediction for inter-prediction, 4×4 discrete sine transform (DST), and sample adaptive offset (SAO) for a deblocking filter.

In this paper, we propose a hardware interpolation architecture for inter-prediction. In H.264/AVC, six-tap and linear filters are used for luma interpolation. However, unlike H.264/AVC, the interpolation in HEVC adopts one-dimensional eight-tap and seven-tap interpolation [6]. Therefore, in the worst case, 11×11 reference pixels are used for a 4×4 luma interpolation. This increases the design complexity and the gate area when it is implemented into the hardware architecture.

In order to achieve high performance, we adopt a highly parallel architecture, which can increase the area. However, we also propose a shared operation processing element (PE) to reduce the area.

The rest of the paper is organized as follows. In Section II, the basic interpolation algorithm is briefly introduced. In Section III, the proposed architecture is presented. Section IV provides the final analysis and the conclusion is

Received 14 December 2012, **Revised** 06 March 2013, **Accepted** 18 March 2013
***Corresponding Author** Kwangki Ryoo (E-mail: kkryoo@gmail.com, Tel: +82-42-821-1710)
Graduate School of Information and Communication, Hanbat National University, 125 Dongseo-daero, Yuseong-gu, Daejeon 305-719, Korea.

Open Access http://dx.doi.org/10.6109/jicce.2013.11.2.118 print ISSN: 2234-8255 online ISSN: 2234-8883

addressed in Section V.

II. BASIC ALGORITHM

Like H.264/AVC, the accuracy of motion compensation in HECV is 1/4 pel for luma samples. To obtain the non-integer luma samples, eight-tap and seven-tap interpolation filters are applied horizontally and vertically to generate luma half-pel and quarter-pel samples, respectively [7].

The fractional positions for HEVC inter-prediction luma interpolation are shown in Fig. 1, which is from HEVC draft 6 [1]. Capital letters in Fig. 1 indicate the integer samples used as reference pixels. Table 1 shows the coefficients used in the luma interpolation filters.

As shown in Table 1, the coefficients can be divided into three types. The a, d, e, f, and g positions use A type coefficients, while the b, h, i, j, and k positions use B type coefficients, and the c, n, p, q, and r positions use C type coefficients. The a, b, and c positions are filtered by applying horizontal filters and the d, h, and n positions are filtered by applying vertical filters. For calculating the other pixel values, both horizontal and vertical filters are used. For example, if the position we have to calculate is f, j, or q, vertical filters are used with the b position as the reference pixel. However, the b position should be calculated by applying horizontal filters in advance.

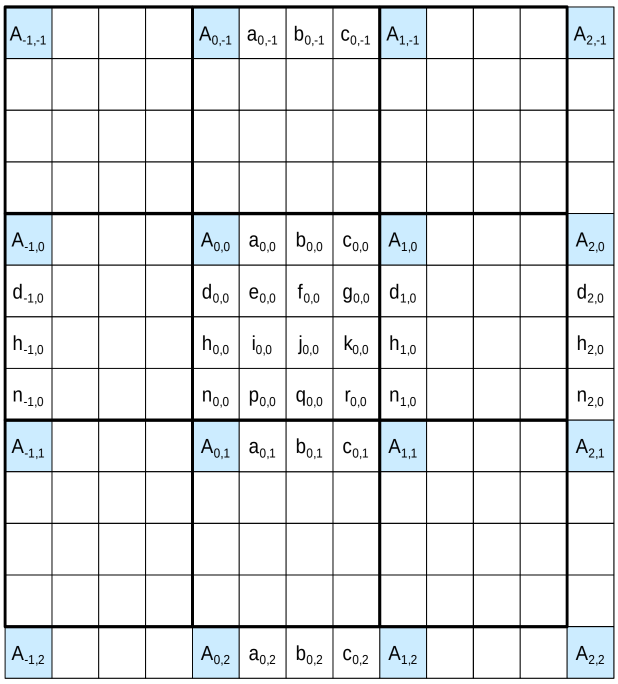
$A_{-1,-1}$ $A_{0,-1}$ $a_{0,-1}$ $b_{0,-1}$ $c_{0,-1}$ $A_{1,-1}$ $A_{2,-1}$
$A_{-1,0}$ $A_{0,0}$ $a_{0,0}$ $b_{0,0}$ $c_{0,0}$ $A_{1,0}$ $A_{2,0}$
$d_{-1,0}$ $d_{0,0}$ $e_{0,0}$ $f_{0,0}$ $g_{0,0}$ $d_{1,0}$ $d_{2,0}$
$h_{-1,0}$ $h_{0,0}$ $i_{0,0}$ $j_{0,0}$ $k_{0,0}$ $h_{1,0}$ $h_{2,0}$
$n_{-1,0}$ $n_{0,0}$ $p_{0,0}$ $q_{0,0}$ $r_{0,0}$ $n_{1,0}$ $n_{2,0}$
$A_{-1,1}$ $A_{0,1}$ $a_{0,1}$ $b_{0,1}$ $c_{0,1}$ $A_{1,1}$ $A_{2,1}$
$A_{-1,2}$ $A_{0,2}$ $a_{0,2}$ $b_{0,2}$ $c_{0,2}$ $A_{1,2}$ $A_{2,2}$

Fig. 1. Integer samples (shaded blocks with upper-case letters) and fractional sample positions (un-shaded blocks with lower-case letters) for quarter sample luma interpolation.

Table 1. Luma interpolation coefficients

Type	Coefficients	Position
A	-1, 4, -10, 58, 17, -5, 1	a, d, e, f, g
B	-1, 4, -11, 40, 40, -11, 4, -1	b, h, i, j, k
C	1, -5, 17, 58, -10, 4, -1	c, n, p, q, r

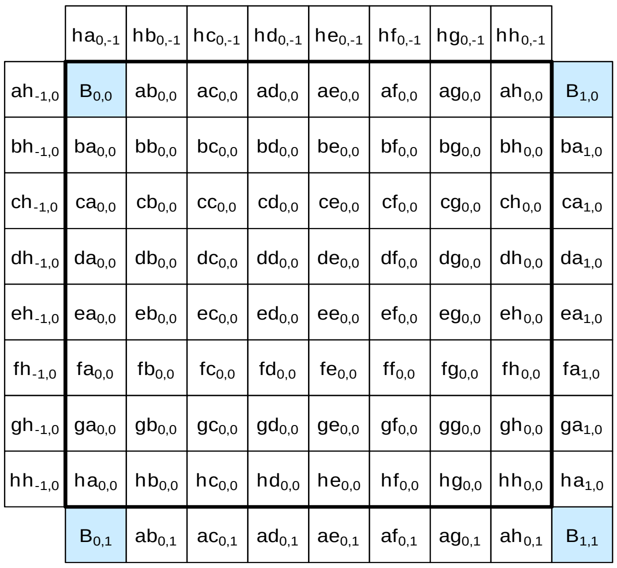
$ha_{0,-1}$ $hb_{0,-1}$ $hc_{0,-1}$ $hd_{0,-1}$ $he_{0,-1}$ $hf_{0,-1}$ $hg_{0,-1}$ $hh_{0,-1}$
$ah_{-1,0}$ $B_{0,0}$ $ab_{0,0}$ $ac_{0,0}$ $ad_{0,0}$ $ae_{0,0}$ $af_{0,0}$ $ag_{0,0}$ $ah_{0,0}$ $B_{1,0}$
$bh_{-1,0}$ $ba_{0,0}$ $bb_{0,0}$ $bc_{0,0}$ $bd_{0,0}$ $be_{0,0}$ $bf_{0,0}$ $bg_{0,0}$ $bh_{0,0}$ $ba_{1,0}$
$ch_{-1,0}$ $ca_{0,0}$ $cb_{0,0}$ $cc_{0,0}$ $cd_{0,0}$ $ce_{0,0}$ $cf_{0,0}$ $cg_{0,0}$ $ch_{0,0}$ $ca_{1,0}$
$dh_{-1,0}$ $da_{0,0}$ $db_{0,0}$ $dc_{0,0}$ $dd_{0,0}$ $de_{0,0}$ $df_{0,0}$ $dg_{0,0}$ $dh_{0,0}$ $da_{1,0}$
$eh_{-1,0}$ $ea_{0,0}$ $eb_{0,0}$ $ec_{0,0}$ $ed_{0,0}$ $ee_{0,0}$ $ef_{0,0}$ $eg_{0,0}$ $eh_{0,0}$ $ea_{1,0}$
$fh_{-1,0}$ $fa_{0,0}$ $fb_{0,0}$ $fc_{0,0}$ $fd_{0,0}$ $fe_{0,0}$ $ff_{0,0}$ $fg_{0,0}$ $fh_{0,0}$ $fa_{1,0}$
$gh_{-1,0}$ $ga_{0,0}$ $gb_{0,0}$ $gc_{0,0}$ $gd_{0,0}$ $ge_{0,0}$ $gf_{0,0}$ $gg_{0,0}$ $gh_{0,0}$ $ga_{1,0}$
$hh_{-1,0}$ $ha_{0,0}$ $hb_{0,0}$ $hc_{0,0}$ $hd_{0,0}$ $he_{0,0}$ $hf_{0,0}$ $hg_{0,0}$ $hh_{0,0}$ $ha_{1,0}$
$B_{0,1}$ $ab_{0,1}$ $ac_{0,1}$ $ad_{0,1}$ $ae_{0,1}$ $af_{0,1}$ $ag_{0,1}$ $ah_{0,1}$ $B_{1,1}$

Fig. 2. Integer samples (shaded blocks with uppercase letters) and fractional sample positions (un-shaded blocks with lowercase letters) for the eighth sample chroma interpolation.

Fig. 2 shows the chroma fractional positions for HEVC inter-prediction interpolation. For the positions aX (X represents the letters from b to h) and Xa shown in Fig. 2 from HEVC draft 6 [1], horizontal filters and vertical filters are used for interpolation, respectively. For the other fractional positions, both horizontal filters and vertical filters must be used, as is done in the luma pixel filtering process. For example, the Xb positions should be calculated by applying vertical filters with the ab position as the reference pixel and the ab position should be calculated by applying a horizontal filter beforehand. Table 2 shows the coefficients used in the chroma interpolation filters.

Table 2. Chroma interpolation coefficients

Type	Coefficient	Position
A	-2, 58, 10, -2	ab, ba, bX
B	-4, 54, 16, -2	ac, ca, cX
C	-6, 46, 28, -4	ad, da, dX
D	-4, 36, 36, -4	ae, ea, eX
E	-4, 28, 46, -6	af, fa, fX
F	-2, 16, 54, -4	ag, ga, gX
G	-2, 10, 58, -2	ah, ha, hX

From Table 2 we can find out there are seven types of coefficients in chroma interpolation and the positions use the corresponding types of coefficients are also shown in Table 2. For example, the ab, ba, and bX positions use the A type filter with the coefficient of {-2, 58, 10, -2}.

III. PROPOSED ARCHITECTURE

In this section, we present the proposed interpolation architecture used for interpolation of HEVC inter-prediction.

A. Proposed PE

There are three types of coefficients in Table 1. However, we determined that if the order of the coefficients in the A type is reversed, the C type coefficients can be obtained. Therefore, if the order of input reference pixels is changed, the same hardware architecture can be used for both the A type and C type. In this way, we designed the A type and B type filters to form basic luma PEs. In the same way, we designed the A type, B type, C type, and D type to contain basic chroma PEs. Eqs. (1) and (2) show the original A type and B type luma filters.

$$AtypeL = -P_0 + 4\times P_1 - 10\times P_2 + 58\times P_3 + 17\times P_4 - 5\times P_5 + P_6\,. \quad (1)$$

$$BtypeL = -P_0 + 4\times P_1 - 11\times P_2 + 40\times P_3 + 40\times P_4 - 11\times P_5 + 4\times P_6 - P_7\,. \quad (2)$$

To eliminate the multiply operation in Eqs. (1) and (2), we converted them into Eqs. (3) and (4).

$$\begin{aligned} AtypeL = &-(P_0+P_5)+(P_4+P_6)-(P_2+P_3) << 1 \\ &+(P_1-(P_2<<1))<<2-(P_3+P_5)<<2 \\ &+(P_4+(P_3<<2))<<4\,. \end{aligned} \quad (3)$$

$$\begin{aligned} BtypeL = &-(P_0+P_7)+(P_1+P_6)-(P_2+P_5) \\ &+((P3+P4)<<2-(P2+P5))<<1 \\ &+((P3+P4)<<2-(P2+P5))<<3\,. \end{aligned} \quad (4)$$

The luma A type PE and B type PE designed from Eqs. (3) and (4) are shown in Figs. 3 and 4.

Figs. 3 and 4 show that both the A type PE and B type PE consist of shift and additional operations. There are 11 and 9 adders in the A type PE and B type PE, respectively. To reduce the critical path, we have inserted registers so that we can boost the maximum frequency.

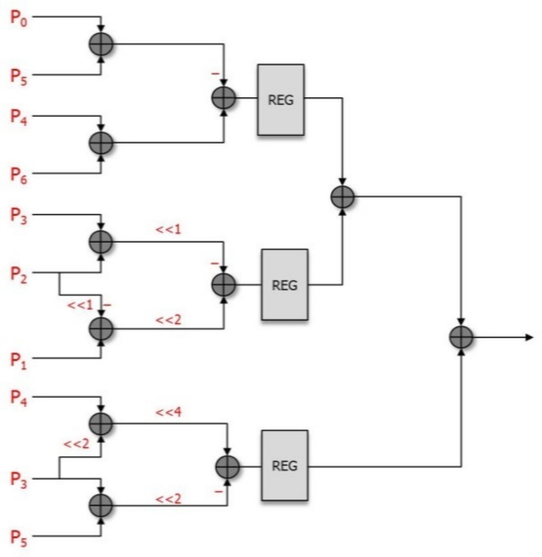

Fig. 3. Luma A type processing element.

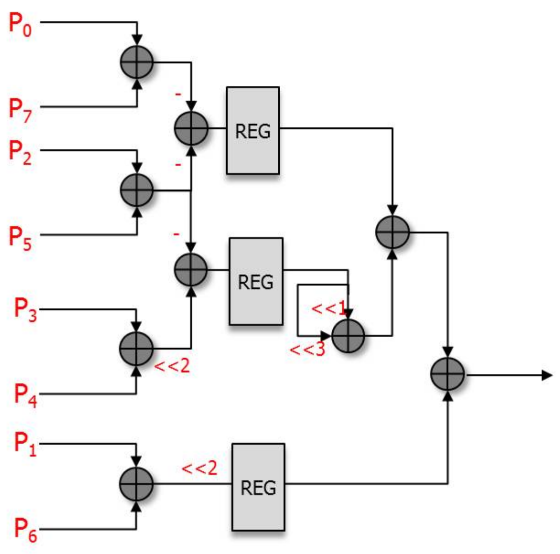

Fig. 4. Luma B type processing element.

The hardware architecture that we propose introduces highly parallel architecture, which increases the gate count. In order to reduce the gate count, the separated A type PE and B type PE are combined into one shared operation filter that comprise a PE. The reason that they can be combined is that they are not used at the same time in the process of filtering.

We determined the common part from Eqs. (3) and (4) for shared operations.

$$\begin{aligned} SOP = &-P_0 + P_1 << 2 + (P_3+P_4) << 4 \\ &+(-P_2-P_5)<<3+(-P_2-P_5)<<1\,. \end{aligned} \quad (5)$$

The shared operation (SOP) in Eq. (5) shows the common part used in both the A type filter and B type filter.

This SOP is used in Eqs. (6) and (7).

$$AtypeL = SOP + P_5 << 2 + P_5 + P_3 << 3 + P_3 << 1 + P_4 + P_6 \,. \tag{6}$$

$$BtypeL = SOP - (P_2 + P_5) + (P_3 + P_4) << 5 + P_6 << 2 - P_7 \,. \tag{7}$$

The original separated A type PE and B type PE have 20 adders in total. However, the combined filter with shared operation has 17 adders, which is 15% less than the previous ones in terms of additional operators.

For the chroma filter, a similar algorithm was applied. The equations for the chroma filter are shown below.

$$AtypeC = -(P_0 + P_3) << 1 + (P_1 + P_2) << 3 + (P_1 + P_2) + P_1 << 5 + P_1 << 4 \,. \tag{8}$$

$$BtypeC = -(P_0 + P_3) << 1 + (P_1 + P_2) << 4 + P_1 << 5 + P_1 << 2 + (P_1 + P_3) << 1 \,. \tag{9}$$

$$CtypeC = -(P_0 + P_3) << 2 + (P_1 + P_2) << 5 + P_1 << 3 + (P_1 - P_2) << 2 + (P_1 - P_0) << 1 \,. \tag{10}$$

$$DtypeC = -(P_0 + P_3) << 2 + (P_1 + P_2) << 5 + (P_1 + P_2) << 2 \,. \tag{11}$$

The equations, from Eqs. (8) to (11), show the chroma filters applied in the proposed architecture. In the chroma filter equations, the P0+P3 part and P1+P2 part can be used as shared operations.

Table 3 shows the comparison of the operators used in several designs. The number of the operators shown in Table 3 is the sum of the operators in each luma PE and chroma PE. Draft [1] contains a large number of multiplier operators, and this greatly increases the gate count. However, the original design and the proposed design consist of adders and shifters to reduce the gate count. Figs. 5 and 6 show the proposed interpolation luma PE and chroma PE applied to the equations from Eqs. (6) to (11).

Table 3. Comparison of the number of operators

Design	Adder	Multiplier	Shifter
Draft [1]	40	44	-
Original	47	-	27
Proposed	37	-	27

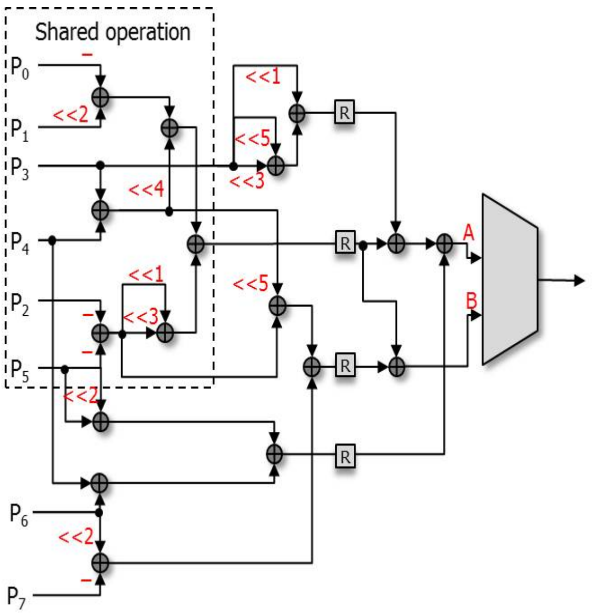

Fig. 5. Luma processing element with shared operation.

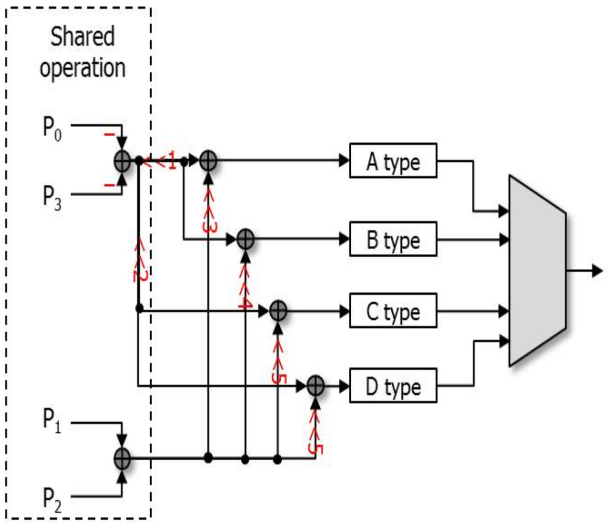

Fig. 6. Chroma processing element with shared operation.

Compared with the original PEs, the architecture shown in Figs. 5 and 6 can reduce the gate count by about 10%.

B. Proposed Interpolator

The proposed PEs shown in Figs. 5 and 6 can only predict one pixel value at a time. However, this does not meet the purpose of decoding 2560×1600 sized videos, which we used for the real-time test. Therefore, we decided to design a highly parallel architecture.

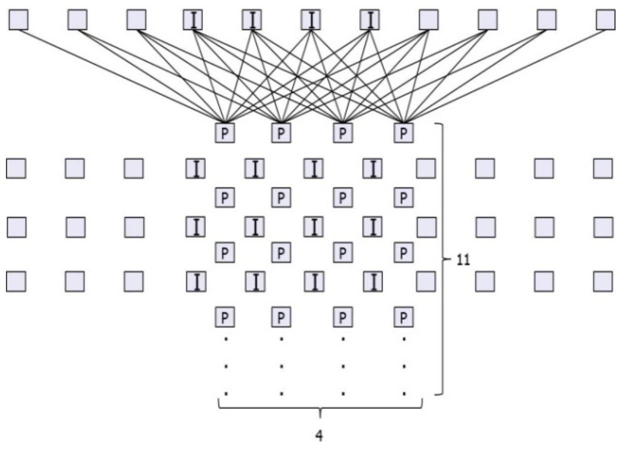

Fig. 7. Proposed luma filter.

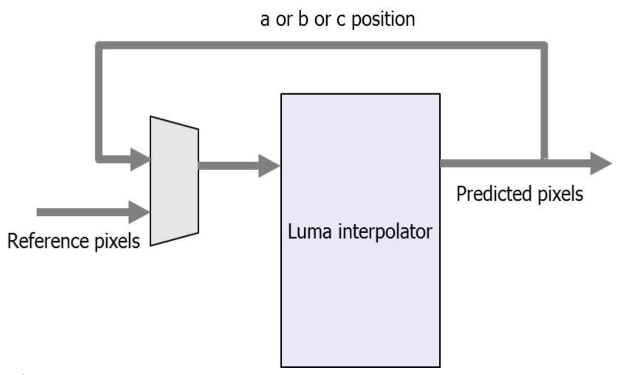

Fig. 8. Luma filter reuse.

Fig. 7 shows the proposed parallel luma filter. The I positions in Fig. 7 show the integer positions in a 4×4 luma block. The P positions indicate the PEs in the proposed filter. There are 11 reference pixels since there are 8 reference pixels needed for each predicted pixel in the worst case, and there are 4 predicted positions in both directions. That means, for a luma 4×4 block, 11×11 reference pixels are needed. There are 4×11 PEs in total, and the luma filter can predict the a, b, c, d, h, and n positions in 2 cycles. In this case, the PEs from the 4th to 7th rows are used. In the other positions, all of the 4×11 PEs are used in the first two cycles and the PEs from the 4th to 7th rows are reused in the next two cycles.

Fig. 8 shows the luma filter reuse block diagram. In the first two cycles, position a, b, or c is predicted, and the predicted pixels feed back to the input of the luma interpolator, as shown in Fig. 8. The chroma filter was designed by a similar method. For a chroma 2×2 block, 5×5 reference pixels are needed. There are 2×5 chroma PEs used as both horizontal and vertical filters. The chroma filter needs 1–2 cycles to predict a chroma 2×2 block.

IV. DESIGN EVALUATION

The proposed architecture was synthesized with the MagnaChip 0.18 μm library and can reach a frequency of 200 MHz. It has to achieve the processing of 2560×1600×30+2560×1600×30×0.5 = 184320000 pixels in one second. In the worst case, for a luma 4×4 block and two chroma 2×2 blocks with bi-directional prediction, 4×2 = 8 cycles and 2×2+2×2 = 8 cycles are needed, respectively. Thus, 16 cycles in total are required to predict 16+8 = 24 pixels. Since it can process 24/16 = 1.5 pixels/cycle on average, and the minimum frequency needed is about 184320000/1.5 = 122 MHz, the proposed design, which has a higher frequency, can decode a 2560×1600 sized video in real-time.

For the luma interpolator, in the best case, it takes 2×2×16 = 64 cycles for one 16×16 block and the process of the pixel for each cycle is 256/64 = 4 pixel/cycle. The gate area of the proposed design is 84400, so the area efficiency is 4/84400 = 0.047 (pixel/cycle)/kgate.

Table 4 shows the comparison of the synthesis result with previous designs. Zheng et al. [8] was designed with three different standards, so it is not accurate to compare the gate count. However, in terms of the number of cycles, the proposed design improved the performance by about 86.67%.

The design of Guo et al. [9] was based on HEVC draft 3, which has different coefficients from the working draft 6 we have applied, and it only implemented a luma interpolator. The proposed design has improved the performance by about 63.64% in terms of the cycle number compared with the design of Guo et al. [9]. Even though the gate area has been greatly increased, the area efficiency is better than that of Guo et al. [9] in the best case.

Table 4. Comparison of the synthesis results

Design	Ref. [8]	Ref. [9]	Original	Proposed
Standard	MPEG-2, H.264, AVS	HEVC draft 3	HEVC draft 6	HEVC draft 6
Luma or chroma	Luma, Chroma	Luma	Luma, chroma	Luma, chroma
Technology	0.18 μm	90 nm	0.18 μm	0.18 μm
Clock frequency (MHz)	148.5	171	200	200
Gate count	21569	32496	93433	84400
Execution cycles per 16×16 block	600	176	80	80
Area efficiency (pixel/cycle)/kgate	-	0.045	Best case, 0.043	Best case 0.047

MPEG: Moving Picture Experts Group, AVS: Audio Video Standard, HEVC: high efficiency video coding.

V. CONCLUSIONS

In this paper, we proposed a highly parallel interpolation hardware architecture for both luma pixels and chroma pixels. It was designed with Verilog HDL and tested with the test vectors extracted from HM7.0. It was synthesized with the design compiler supported by the IC Design Education Center (IDEC). The synthesis results show the maximum frequency of the proposed design can reach up to 200 MHz. The experimental results showed that the proposed design improved the performance by about 75.15%, with an area efficiency better than previous designs in the best case.

ACKNOWLEDGMENTS

This research was financially supported by the Ministry of Education, Science and Technology (MEST) and the National Research Foundation of Korea (NRF) through the Human Resource Training Project for Regional Innovation. This work was also sponsored by the ETRI SW-SoC R&BD Center, Human Resource Development Project.

REFERENCES

[1] B. Bross, W. J. Han, G. J. Sullivan, J. R. Ohm, and T. Wiegand, "High efficiency video coding (HEVC) text specification draft 6," The Joint Collaborative Team on Video Coding (JCT-VC), *JCTVC-H1003*, 2012.

[2] International Telecommunication Union, "Advanced video coding for generic audiovisual services," International Telecommunication Union, Geneva, Switzerland, *ITU-T Recommendation H.264 and ISO/IEC 14496-10*, 2003.

[3] D. Zhou and P. Liu, "A hardware-efficient dual-standard VLSI architecture for MC interpolation in AVS and H.264," in *IEEE International Symposium on Circuits and Systems*, New Orleans, LA, pp. 2910-2913, 2007.

[4] Y. J. Kim, "Fast intra-prediction mode decision algorithm for H.264/AVC using non-parametric thresholds and simplified directional masks," *International Journal of Maritime Information and Communication Sciences*, vol. 7, no. 4, pp. 501-506, 2009.

[5] Y. J. Kim, "A non-parametric fast block size decision algorithm for H.264/AVC intra prediction," *International Journal of Maritime Information and Communication Sciences*, vol. 7, no. 2, pp. 193-198, 2009.

[6] E. Alshina, A. Alshin, J. H. Park, J. Lou, and K. Minoo, "CE3: 7 taps interpolation filters for quarter pel position MC from Samsung and Motorola mobility," The Joint Collaborative Team on Video Coding (JCT-VC), *JCTVC-G778*, 2011.

[7] M. T. Pourazad, C. Doutre, M. Azimi, and P. Nasiopoulos, "HEVC: the new gold standard for video compression: How Does HEVC Compare with H.264/AVC?," *IEEE Consumer Electronics Magazine*, vol. 1, no. 3, pp. 36-46, 2012.

[8] J. Zheng, W. Gao, D. Wu, and D. Xie, "A novel VLSI architecture of motion compensation for multiple standards," *IEEE Transactions on Consumer Electronics*, vol. 54, no. 2, pp. 687-694, May 2008.

[9] Z. Guo, D. Zhou, and S. Goto, "An optimized MC interpolation architecture for HEVC," in *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing*, Kyoto, Japan, pp. 1117-1120, 2012.

Xianzhe Jin
was born in Yanji, China in 1982. He received his B.S. degree in Information and Communication Engineering from Paichai University, Daejeon, South Korea in 2007 and his M.S. degree in Information and Communication Engineering from Hanbat National University in 2009. He is currently a Ph.D. candidate in Information and Communication Engineering at Hanbat National University. His current research interests include SoC platform design and verification, hardware/software co-design, and video CODEC design.

Kwangki Ryoo
is an Associate Professor of the Department of Information and Communication Engineering at Hanbat National University. He was born in Gongju, South Korea in 1964. He studied at Hanyang University, Seoul, South Korea, where he received his B.S., M.S., and Ph.D. degrees in Electronic Engineering in 1986, 1988, and 2000, respectively. From 1991 to 1994, he was a full-time lecturer in the department of Electronic Engineering at the Korea Military Academy (KMA), Seoul, South Korea. From 2000 to 2002, he worked in the System IC Design Team at the Electronics and Telecommunications Research Institute (ETRI), Daejeon, South Korea. From 2010 to 2011, he was a visiting scholar at the University of Texas at Dallas. His current research interests include engineering education and SoC design for multimedia CODECs.